Integrating Cache Related Pre-emption Delay Analysis into EDF Scheduling

Will Lunniss[1] Sebastian Altmeyer[2] Claire Maiza[3] Robert I. Davis[1]

[1]Real-Time Systems Research Group, University of York, UK

{wl510, rob.davis}@york.ac.uk

[2]Department of Computer Science, Saarland University, Germany

altmeyer@cs.uni-sb.de

[2]Verimag, Grenoble INP, Grenoble, France

Claire.maiza@imag.fr

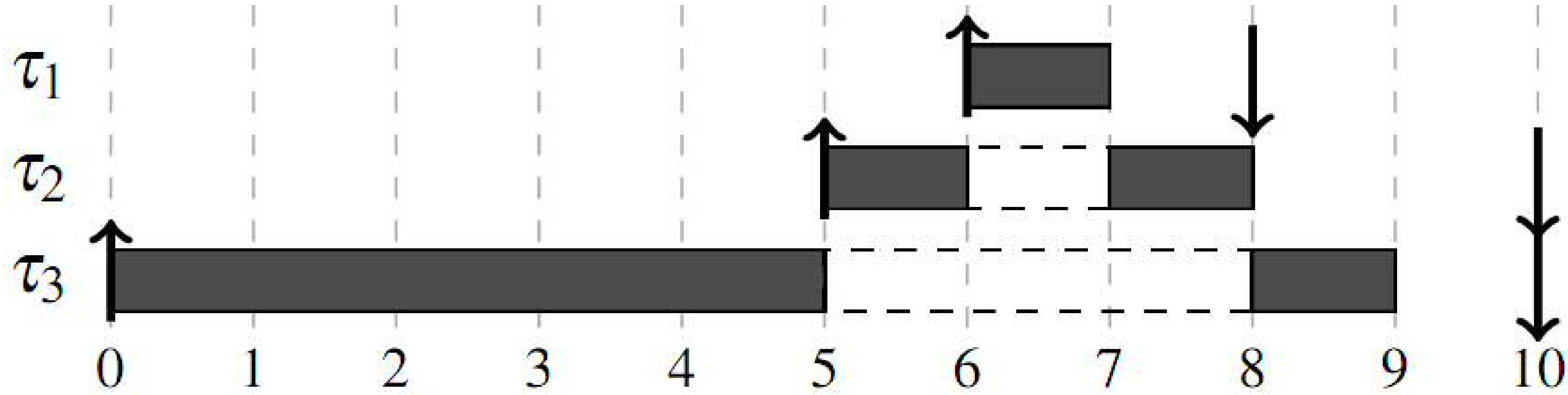
Outline

- EDF Scheduling
- CRPD
- Integrating CRPD analysis into EDF
- Comparison with existing approach
- Improved CRPD analysis
- Case study
- Synthetic taskset evaluation
- Conclusions

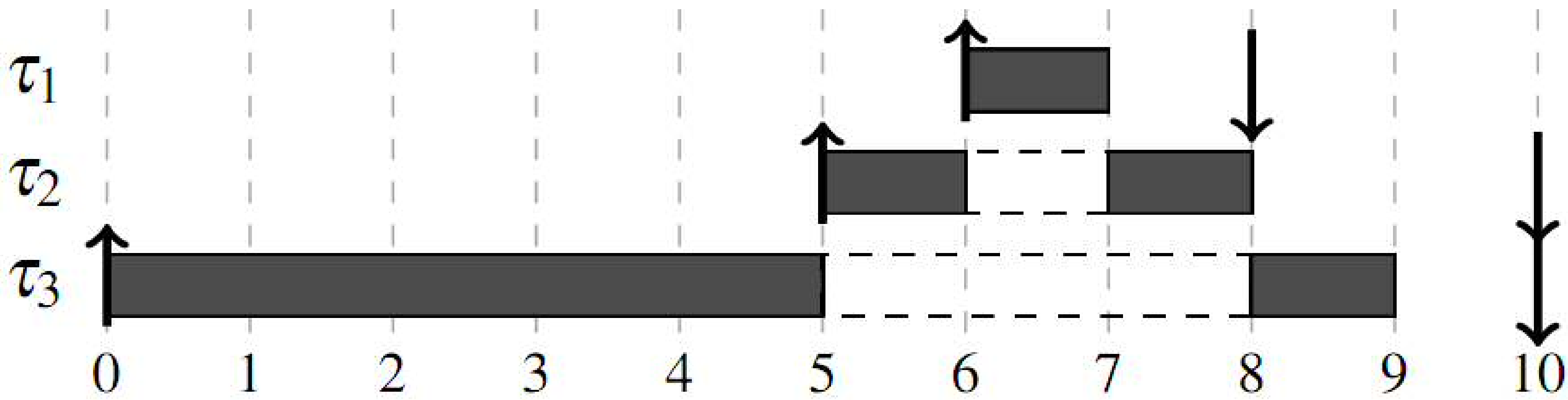
Earliest Deadline First (EDF)

- It is dynamic scheduling algorithm
- Schedules the job of the task with the earliest absolute deadline first
- Proven to be optimal by Dertouzos on a single core processor

Determining which job should run

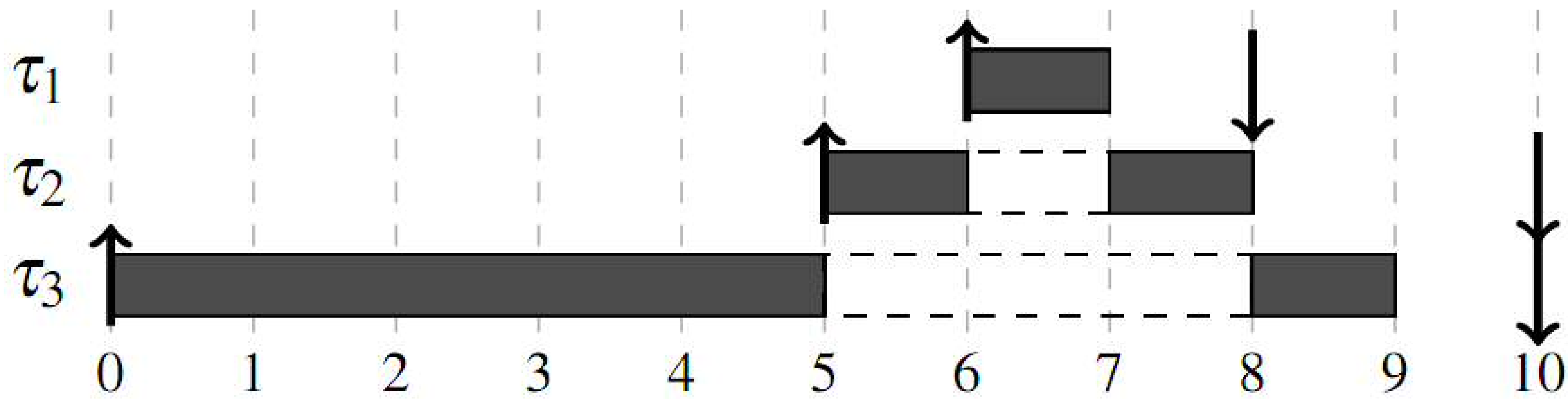


Determining which job should run



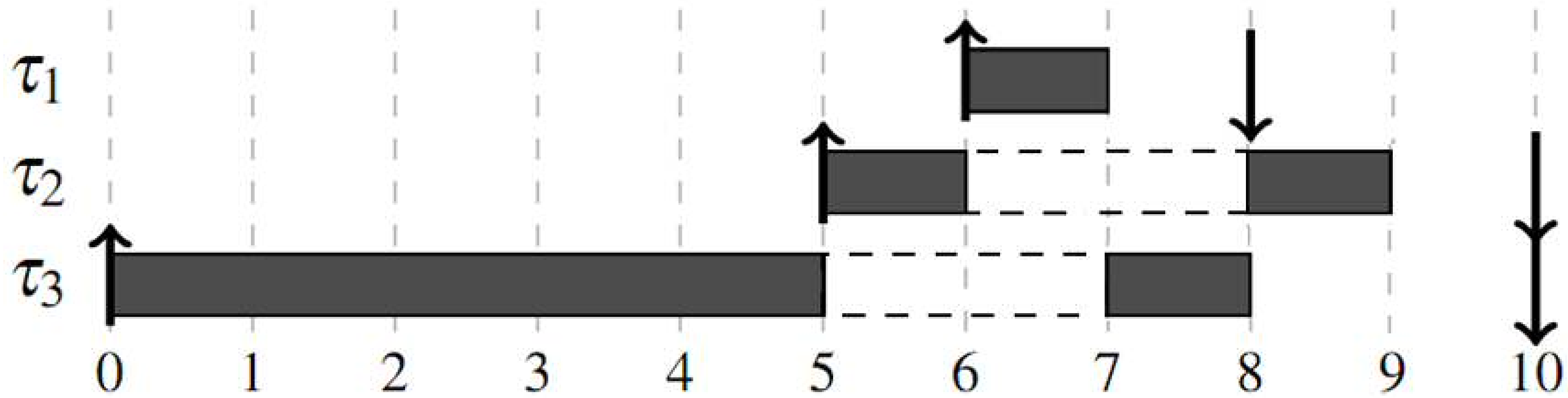
- If two jobs have the same absolute deadline
  - We assume that the job with the lowest task index is chosen
  - E.g. τ_2 pre-empts τ_3 in the above example

Determining which job should run



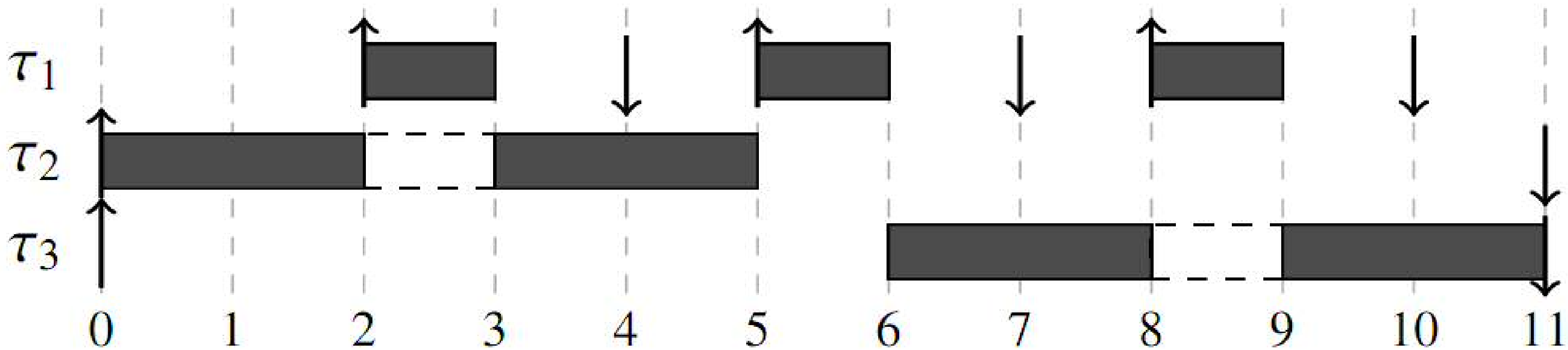
- If two jobs have the same absolute deadline
  - Ensures that two tasks cannot pre-empt each other
  - Ensures that after a pre-emption, the task that was pre-empted last is resumed first
  - E.g τ_2 is resumed at $t = 7$

Determining which job should run



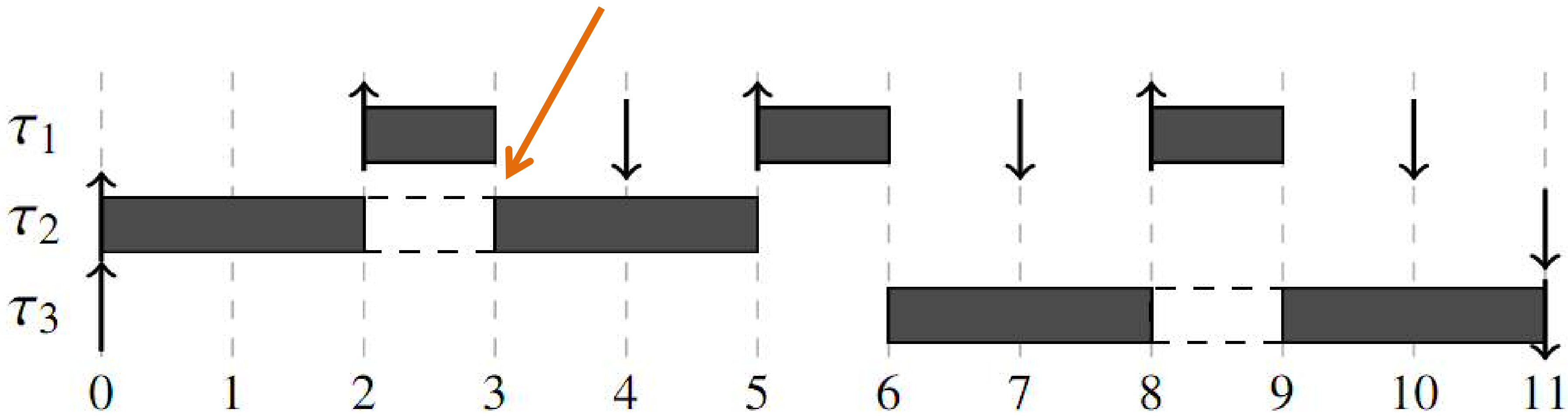
- If two jobs have the same absolute deadline
  - Ensures that two tasks cannot pre-empt each other
  - Ensures that after a pre-emption, the task that was pre-empted last is resumed first
  - E.g τ_2 is resumed at $t = 7$, rather than τ_3

Determining which job should run



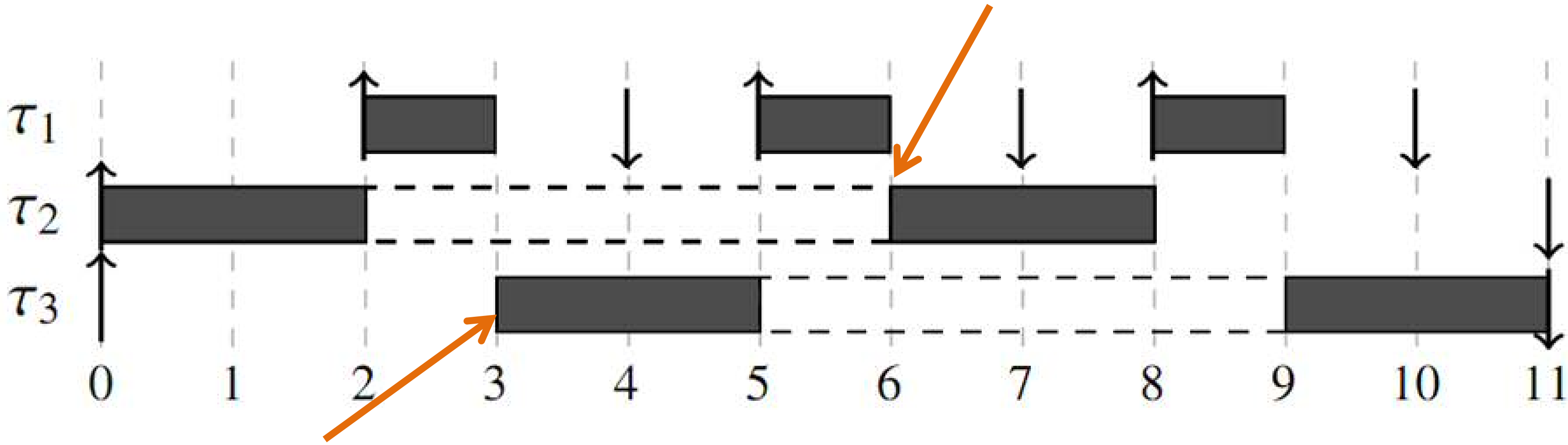
- Also applies for jobs with the same relative deadline and release time
  - E.g. τ_2 is resumed at $t = 3$, rather than τ_3 starting

Determining which job should run



- Also applies for jobs with the same relative deadline and release time
  - E.g. τ_2 is resumed at $t = 3$, rather than τ_3 starting

Determining which job should run



- Also applies for jobs with the same relative deadline and release time
  - E.g. τ_2 is resumed at $t = 3$, rather than τ_3 starting

Schedulability test

- If all tasks have implicit deadlines ($D_i = T_i$), schedulability test is

Schedulability test

- If all tasks have implicit deadlines ($D_i = T_i$), schedulability test is

$$U \leq 1$$

Schedulability test

- If all tasks have implicit deadlines ($D_i = T_i$), schedulability test is

$$U \leq 1$$

- If $D_i \neq T_i$ then the test is still *necessary*, but is no longer *sufficient*

Schedulability test

- If all tasks have implicit deadlines ($D_i = T_i$), schedulability test is

$$U \leq 1$$

- If $D_i \neq T_i$ then the test is still *necessary*, but is no longer *sufficient*
- Need to do another test

Processor demand bound function

$$h(t) =$$

Processor demand bound function

$$h(t) = \sum_{i=1}$$

Sum over each task

Processor demand bound function

$$h(t) = \sum_{i=1} \max\left\{0, 1 + \left\lfloor \frac{t - D_i}{T_i} \right\rfloor\right\}$$

Sum over each task

the number of jobs a task has which are released and have their deadlines in the interval *t*

Processor demand bound function

$$h(t) = \sum_{i=1} \max\left\{0, 1 + \left\lfloor \frac{t - D_i}{T_i} \right\rfloor\right\} C_i$$

Sum over each task

the number of jobs a task has which are released and have their deadlines in the interval *t*

multiplied by the tasks' execution time

Processor demand bound function

$$h(t) = \sum_{i=1} \max\left\{0, 1 + \left\lfloor \frac{t - D_i}{T_i} \right\rfloor\right\} C_i$$

Sum over each task

the number of jobs a task has which are released and have their deadlines in the interval *t*

multiplied by the tasks' execution time

S. K. Baruah, L. E. Rosier, and R. R. Howell, "Algorithms and Complexity Concerning the Preemptive Scheduling of Periodic Real-Time Tasks on One Processor," *Real-Time Systems*, vol. 2, no. 4, pp. 301-324, 1990

Schedulability test

- A taskset is schedulable iff $h(t) \leq t$ for all values of t
  - The execution time requirement must be less than or equal to the available time

Schedulability test

- A taskset is schedulable iff $h(t) \leq t$ for all values of t
  - The execution time requirement must be less than or equal to the available time
- $h(t)$ can only change when t is equal to an absolute deadline
- Bound the maximum value of t, L, using either
  - *Hyper-period:* Least common multiple of task periods
  - Synchronous busy period

Schedulability test

- There are still a large number of values for t that need to be checked

Schedulability test

- There are still a large number of values for t that need to be checked
- Can be reduced by using the *Quick convergence Processor-demand Analysis* (QPA) algorithm by Zhang and Burns
  - Starts with a value of t close to L
  - Iterates back towards 0 checking a significantly smaller number of values

Schedulability test

- There are still a large number of values for *t* that need to be checked
- Can be reduced by using the *Quick convergence Processor-demand Analysis* (QPA) algorithm by Zhang and Burns
  - Starts with a value of *t* close to *L*
  - Iterates back towards 0 checking a significantly smaller number of values

F. Zhang and A. Burns, "Schedulability Analysis for Real-Time Systems with EDF Scheduling," *IEEE Transactions on Computers*, vol. 58, no. 9, pp. 1250-1258, September 2009

Pre-emptions and *Cache Related Pre-empt Delays (CRPD)*

Pre-emptions and *Cache Related Pre-empt Delays (CRPD)*

- Pre-empting task can evict blocks belonging to the pre-empted task
- CRPD are introduced when the pre-empted task has to reload some of those evicted cache blocks after resuming

Cache block categorisation

- *Evicting Cache Blocks* (ECBs)
  - Loaded into cache and can therefore evict other blocks

Cache block categorisation

- *Evicting Cache Blocks* (ECBs)
  – Loaded into cache and can therefore evict other blocks
- *Useful Cache Blocks* (UCBs)
  – Reused once they have been loaded into cache before potentially being evict by the task
  – If evicted by another task, they may have to be reloaded which introduces CRPD
  – UCBs are always ECBs

Cache block categorisation

- Example block classification

Cache block categorisation

- Example block classification

- Instructions inside loops are often UCBs as they get reused

CRPD analysis

- Need to calculate the number of blocks evicted during a pre-emption that must be reloaded
- Multiply by the cost to reload each block, BRT

CRPD analysis

- Need to calculate the number of blocks evicted during a pre-emption that must be reloaded
- Multiply by the cost to reload each block, BRT
- Could take a simple approach and assume every block evicted by a pre-empting task must be re-loaded e.g.

$$\gamma_{t,j} = \mathrm{BRT} \bullet \left| \mathrm{ECB}_j \right|$$

CRPD analysis

- Adapted a number of approaches for FP to work with EDF
- Defined:
  - the sets of tasks which can/cannot pre-empt each other
  - how often these pre-emptions can occur within the interval *t*

CRPD analysis

- Adapted a number of approaches for FP to work with EDF
- Defined:
  - the sets of tasks which can/cannot pre-empt each other
  - how often these pre-emptions can occur within the interval t
- Then include the CRPD into the $h(t)$ calculation

Integrating CRPD analysis into the $h(t)$ calculation

$$h(t) = \sum_{j=1}^{n} \max\left\{0, 1 + \left\lfloor \frac{t - D_j}{T_j} \right\rfloor\right\} C_j$$

Integrating CRPD analysis into the $h(t)$ calculation

Calculate the CRPD caused by one job of task τ_j in the interval t

$$h(t) = \sum_{j=1}^{n} \max\left\{0, 1 + \left\lfloor \frac{t - D_j}{T_j} \right\rfloor\right\}\left(C_j + \gamma_{t,j}\right)$$

Integrating CRPD analysis into the $h(t)$ calculation

Calculate the CRPD caused by one job of task τ_j in the interval t

$$h(t) = \sum_{j=1}^{n} \max\left\{0, 1 + \left\lfloor \frac{t - D_j}{T_j} \right\rfloor\right\} \left(C_j + \gamma_{t,j}\right)$$

Then add it to the execution time of that job of task τ_j

Effect of CRPD on task utilisation and $h(t)$ calculation

- The analysis effectively increases the execution time of a task by the CRPD it causes
- Need to account for this when calculating the utilisation of a task and taskset
- Also need to use this when calculating the upper bound of t used for calculating $h(t)$

Set of pre-empting tasks

- Based on the tasks' relative deadline
  - Assume that any task τ_j with a relative deadline $D_j < D_i$ can pre-empt task τ_i

Set of pre-empting tasks

- Based on the tasks' relative deadline
  - Assume that any task τ_j with a relative deadline $D_j < D_i$ can pre-empt task τ_i
- The set of tasks that can pre-empt task τ_i is:

Set of pre-empting tasks

- Based on the tasks' relative deadline
  - Assume that any task τ_j with a relative deadline $D_j < D_i$ can pre-empt task τ_i
- The set of tasks that can pre-empt task τ_i is:

$$hp(i) = \{\forall \tau_j \mid D_j < D_i\}$$

Set of pre-empted tasks

- Task τ_j can pre-empt any tasks whose relative deadline is greater than it's relative deadline

Set of pre-empted tasks

- Task τ_j can pre-empt any tasks whose relative deadline is greater than it's relative deadline
- Can exclude all tasks whose relative deadlines are greater than t
  - They do not need to be included when calculating $h(t)$, see paper for proof

Set of pre-empted tasks

- Task τ_j can pre-empt any tasks whose relative deadline is greater than it's relative deadline
- Can exclude all tasks whose relative deadlines are greater than t
  - They do not need to be included when calculating $h(t)$, see paper for proof

$$\mathrm{aff}(t, j) = \{\forall \tau_i \mid t \geq D_i > D_j\}$$

UCB-Union

- Based on approach by Tan and Mooney

$$\gamma_{t,j}^{ucb-u} = \mathrm{BRT} \bullet \left| \qquad \right|$$

UCB-Union

- Based on approach by Tan and Mooney

$$\gamma_{t,j}^{ucb-u} = \mathrm{BRT} \bullet \left| \left(\bigcup_{\forall k \in \mathrm{aff}(t,j)} \mathrm{UCB}_k \right) \right|$$

UCB-Union

- Based on approach by Tan and Mooney

$$\gamma_{t,j}^{ucb-u} = \mathrm{BRT} \bullet \left| \left(\bigcup_{\forall k \in \mathrm{aff}(t,j)} \mathrm{UCB}_k \right) \right|$$

Calculate the union of the UCBs of all tasks that:

- can be evicted by the pre-empting task τ_j
- have a job with a release time and absolute deadline within the interval t

UCB-Union

- Based on approach by Tan and Mooney

$$\gamma_{t,j}^{ucb-u} = \mathrm{BRT} \bullet \left| \left(\bigcup_{\forall k \in \mathrm{aff}(t,j)} \mathrm{UCB}_k \right) \cap \mathrm{ECB}_j \right|$$

Calculate the union of the UCBs of all tasks that:

- can be evicted by the pre-empting task τ_j
- have a job with a release time and absolute deadline within the interval t

intersected with the ECBs of the pre-empting task

UCB-Union

- Based on approach by Tan and Mooney

$$\gamma_{t,j}^{ucb-u} = \mathrm{BRT} \bullet \left| \left(\bigcup_{\forall k \in \mathrm{aff}(t,j)} \mathrm{UCB}_k \right) \cap \mathrm{ECB}_j \right|$$

Calculate the union of the UCBs of all tasks that:

- can be evicted by the pre-empting task τ_j
- have a job with a release time and absolute deadline within the interval *t*

intersected with the ECBs of the pre-empting task

Y. Tan and V. Mooney, "Timing Analysis for Preemptive Multitasking Real-Time Systems with Caches," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 6, no. 1, February 2007.

ECB-Union

- Based on the approach by Altmeyer *et al.*

$$\gamma_{t,j}^{ecb-u} = \mathrm{BRT} \bullet$$

ECB-Union

- Based on the approach by Altmeyer *et al.*

$$\gamma_{t,j}^{ecb-u} = \mathrm{BRT} \bullet \left(\bigcup_{h \in hp(j) \cup \{j\}} \mathrm{ECB}_h \right)$$

Assume that task τ_j has already been pre-empted. Include the union of ECBs belonging to all tasks that can pre-empt it

ECB-Union

- Based on the approach by Altmeyer *et al.*

$$\gamma_{t,j}^{ecb-u} = \mathrm{BRT} \bullet \left| \mathrm{UCB}_k \cap \left(\bigcup_{h \in hp(j) \cup \{j\}} \mathrm{ECB}_h \right) \right|$$

Assume that task τ_j has already been pre-empted. Include the union of ECBs belonging to all tasks that can pre-empt it

ECB-Union

- Based on the approach by Altmeyer *et al.*

$$\gamma_{t,j}^{ecb-u} = \mathrm{BRT} \bullet \max_{\forall k \in \mathrm{aff}(t,j)} \left\{ \left| \mathrm{UCB}_k \cap \left(\bigcup_{h \in hp(j) \cup \{j\}} \mathrm{ECB}_h \right) \right| \right\}$$

Calculate the maximum number of UCBs that may need to be reloaded by any task that is directly pre-empted by task τ_j

Assume that task τ_j has already been pre-empted. Include the union of ECBs belonging to all tasks that can pre-empt it

ECB-Union

- Based on the approach by Altmeyer *et al.*

$$\gamma_{t,j}^{ecb-u} = \mathrm{BRT} \bullet \max_{\forall k \in \mathrm{aff}(t,j)} \left\{ \left| \mathrm{UCB}_k \cap \left(\bigcup_{h \in hp(j) \cup \{j\}} \mathrm{ECB}_h \right) \right| \right\}$$

Calculate the maximum number of UCBs that may need to be reloaded by any task that is directly pre-empted by task τ_j

Assume that task τ_j has already been pre-empted. Include the union of ECBs belonging to all tasks that can pre-empt it

S. Altmeyer, R.I. Davis, and C. Maiza, "Cache Related Pre-emption Delay Aware Response Time Analysis for Fixed Priority Pre-emptive Systems," in *Proceedings of the 32nd IEEE Real-Time Systems Symposium (RTSS)*, Vienna, Austria, 2011, pp. 261-271.

Existing approach

- In 2007, Ju *et al.* presented an approach
  - We refer to it as the JCR approach after their initials

$$\gamma_i^{jcr} =$$

Existing approach

- In 2007, Ju *et al.* presented an approach
  - We refer to it as the JCR approach after their initials

$$\gamma_i^{jcr} = \mathrm{BRT} \bullet \sum_{j \in hp(i)}$$

Sum for every task τ_j that can pre-empt task τ_i

Existing approach

- In 2007, Ju *et al.* presented an approach
  - We refer to it as the JCR approach after their initials

$$\gamma_i^{jcr} = \mathrm{BRT} \bullet \sum_{j \in hp(i)} P_j(D_i)$$

Sum for every task τ_j that can pre-empt task τ_i

the number of times task τ_j can pre-empt a single job of task τ_i

Existing approach

- In 2007, Ju *et al.* presented an approach
  - We refer to it as the JCR approach after their initials

$$\gamma_i^{jcr} = \mathrm{BRT} \bullet \sum_{j \in hp(i)} P_j(D_i) \left| \mathrm{UCB}_i \cap \mathrm{ECB}_j \right|$$

Sum for every task τ_j that can pre-empt task τ_i

the number of times task τ_j can pre-empt a single job of task τ_i

multiplied with the number of task τ_i UCBs that could be evicted task τ_j ECBs

Existing approach

- In 2007, Ju *et al.* presented an approach
  - We refer to it as the JCR approach after their initials

$$\gamma_i^{jcr} = \mathrm{BRT} \bullet \sum_{j \in hp(i)} P_j(D_i) \left| \mathrm{UCB}_i \cap \mathrm{ECB}_j \right|$$

Sum for every task τ_j that can pre-empt task τ_i

the number of times task τ_j can pre-empt a single job of task τ_i

multiplied with the number of task τ_i UCBs that could be evicted task τ_j ECBs

Ju, S. Chakraborty, and A. Roychoudhury, "Accounting for Cache-Related Preemption Delay in Dynamic Priority Schedulability Analysis," in *Design, Automation and Test in Europe Conference and Exposition (DATE)*, Nice, France, 2007, pp. 1623-1628

Existing approach

- Can be pessimistic for nested pre-emptions
- Calculates the cost between each pair of tasks

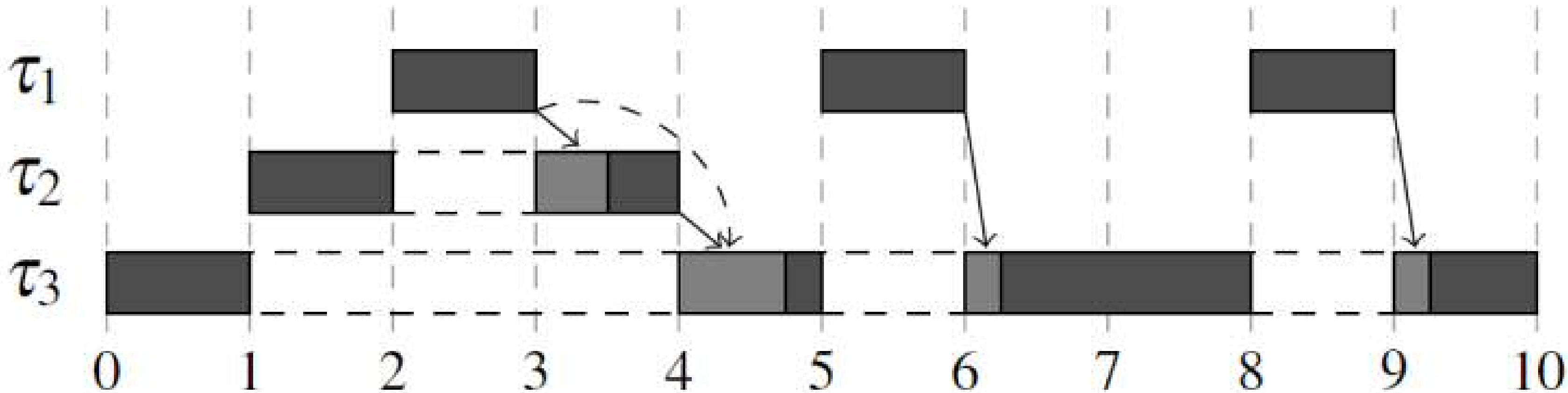
Existing approach

- Can be pessimistic for nested pre-emptions
- Calculates the cost between each pair of tasks
- If pre-empting tasks have ECBs located in the same cache sets, they will be counted multiple times
  – More likely when there the cache utilisation is high

Existing approach

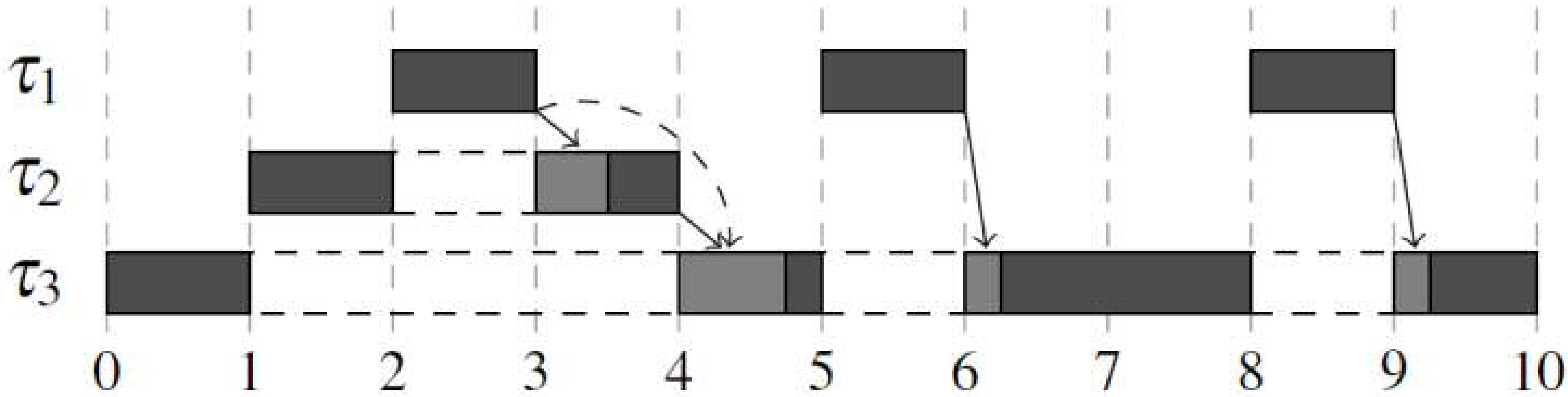
- Can be pessimistic for nested pre-emptions
- Calculates the cost between each pair of tasks
- If pre-empting tasks have ECBs located in the same cache sets, they will be counted multiple times
  - More likely when there the cache utilisation is high
- It is incomparable to the approaches we have presented so far

Improved CRPD analysis



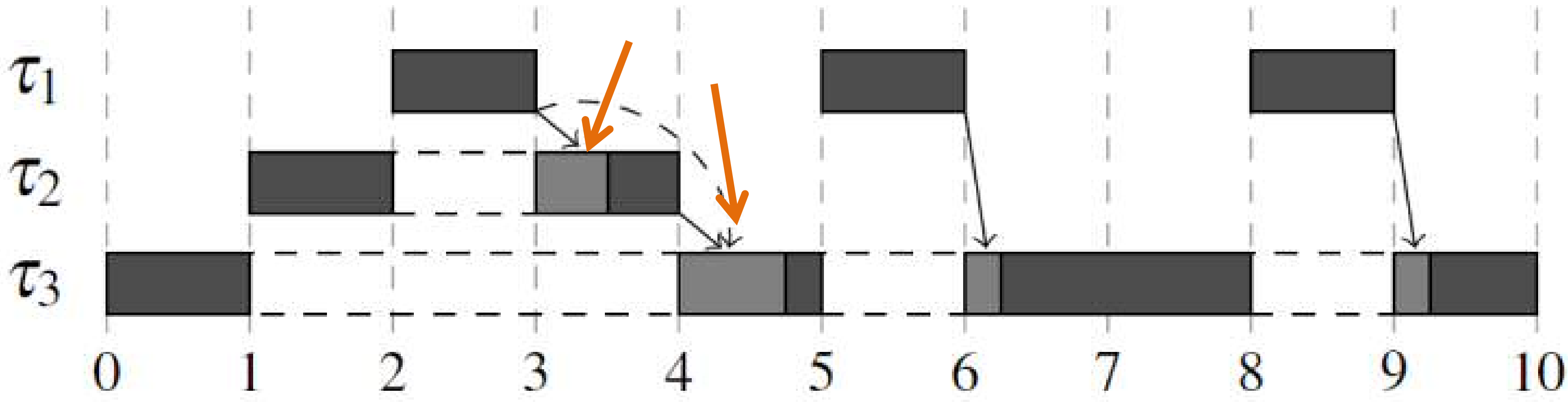
- The UCB-Union and ECB-Union approach can be pessimistic
- They assume intermediate tasks are pre-empted the same number of times as the pre-empted task

Improved CRPD analysis



- E.g. the cost of τ_2 pre-empting task τ_3 is counted three times rather than once

Improved CRPD analysis



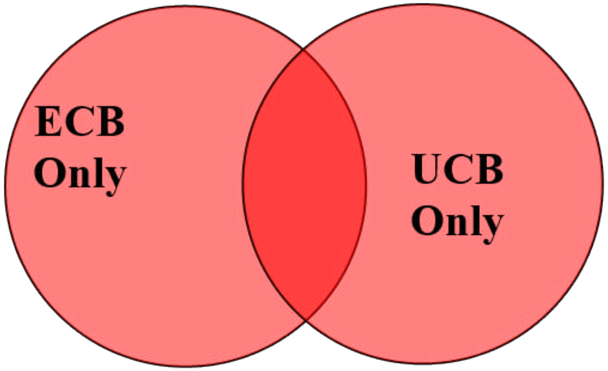
- E.g. the cost of τ_2 pre-empting task τ_3 is counted three times rather than once

Multiset approaches

- ECB-Union Multiset and UCB-Union Multiset
- Factor in the number of times that intermediate tasks pre-empt the pre-empted task to tighten the bound
  - See paper for details

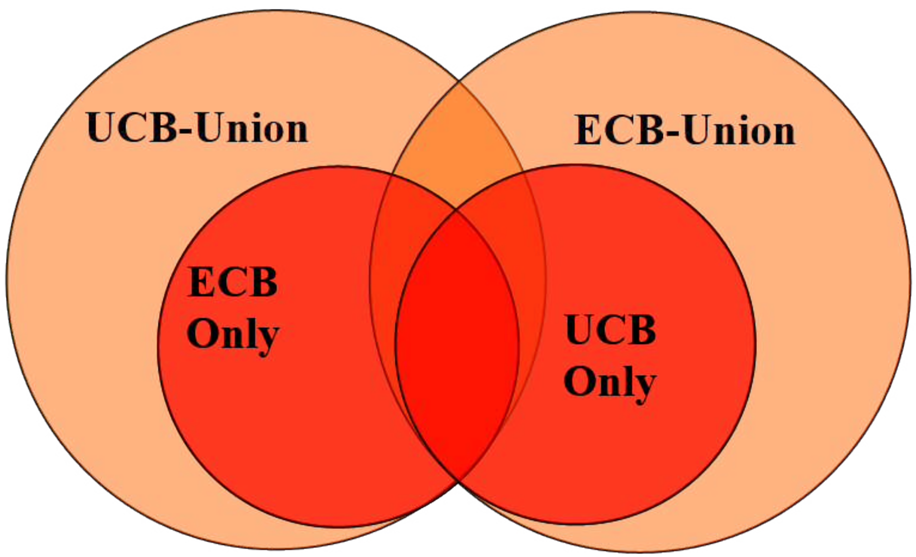
Comparison of approaches

Schedulable Tasksets

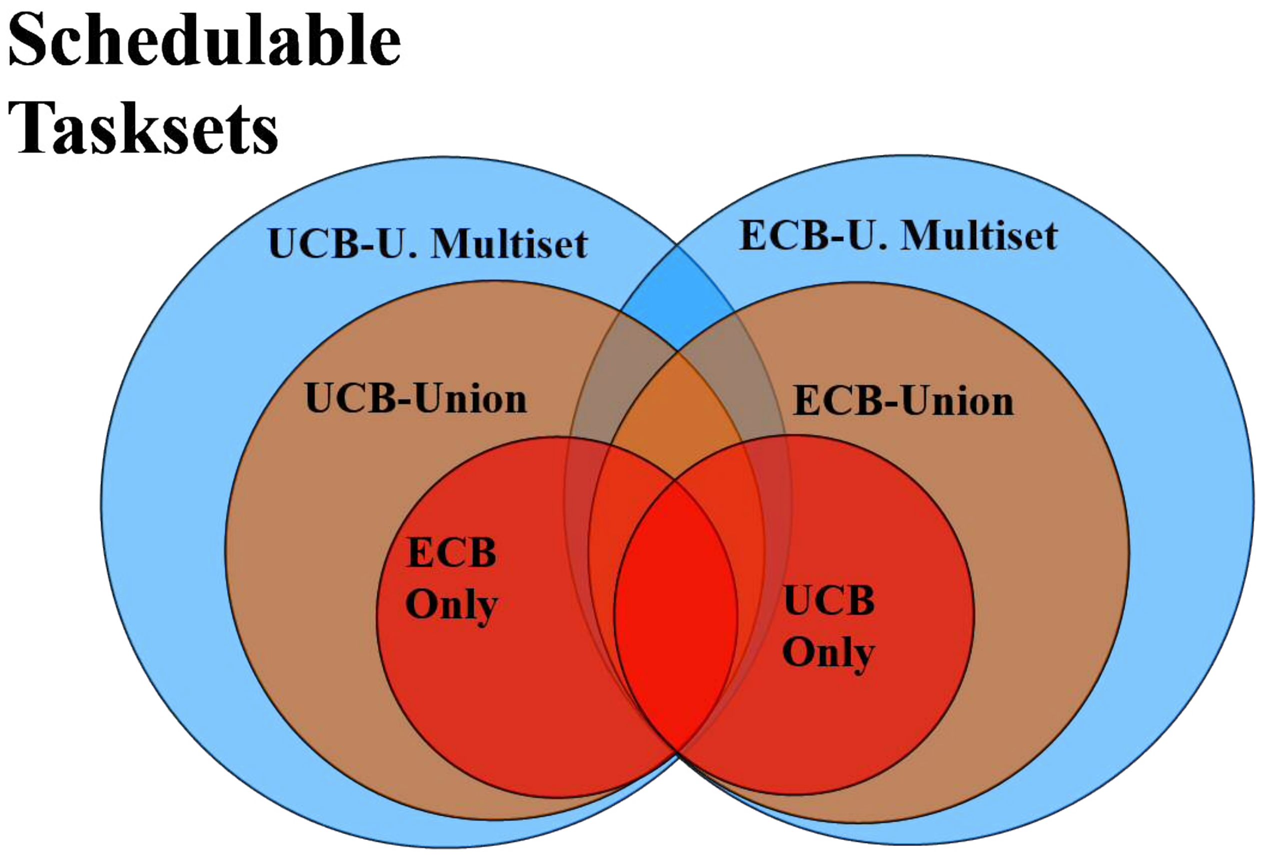


Comparison of approaches

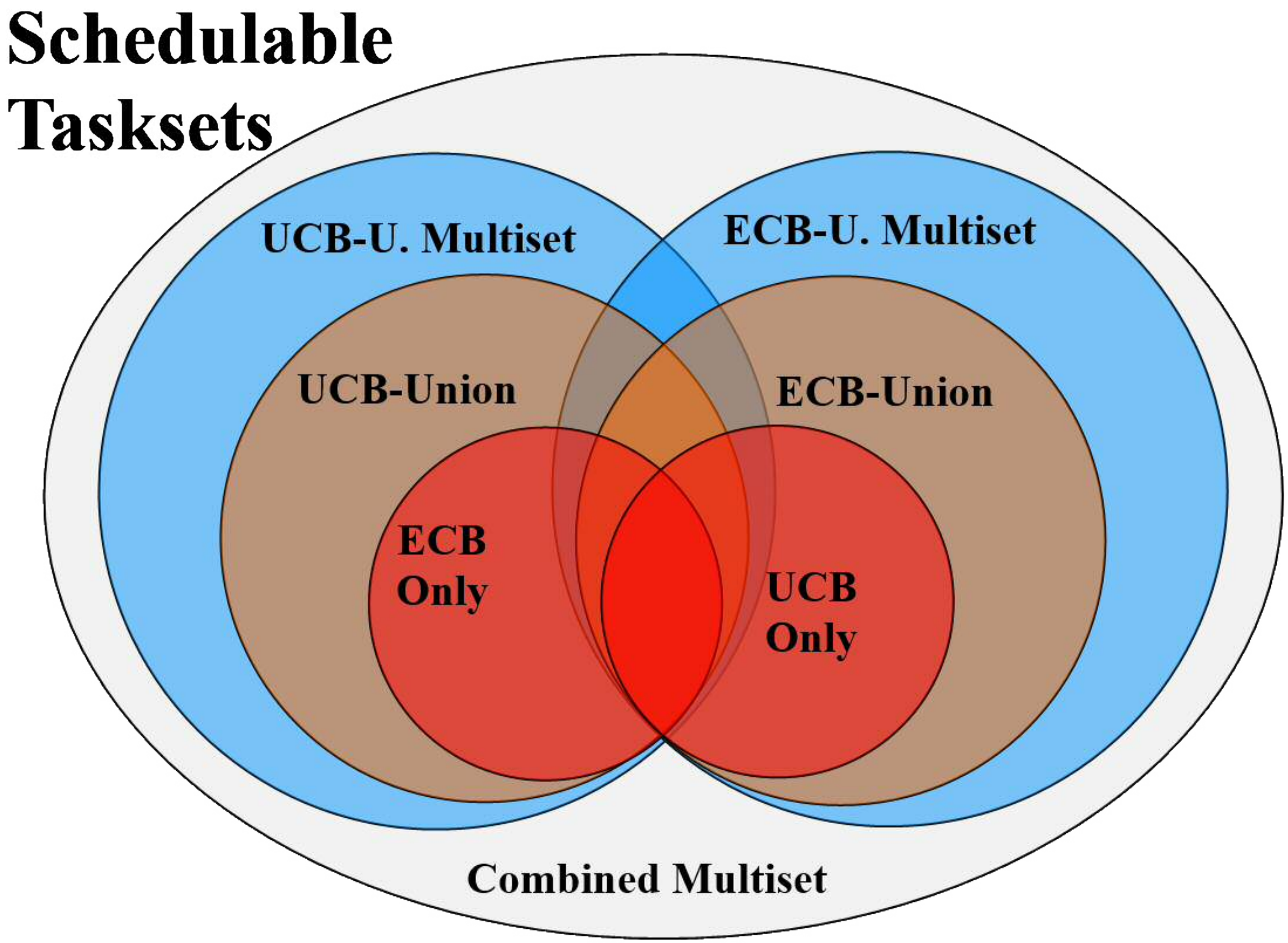
Schedulable Tasksets



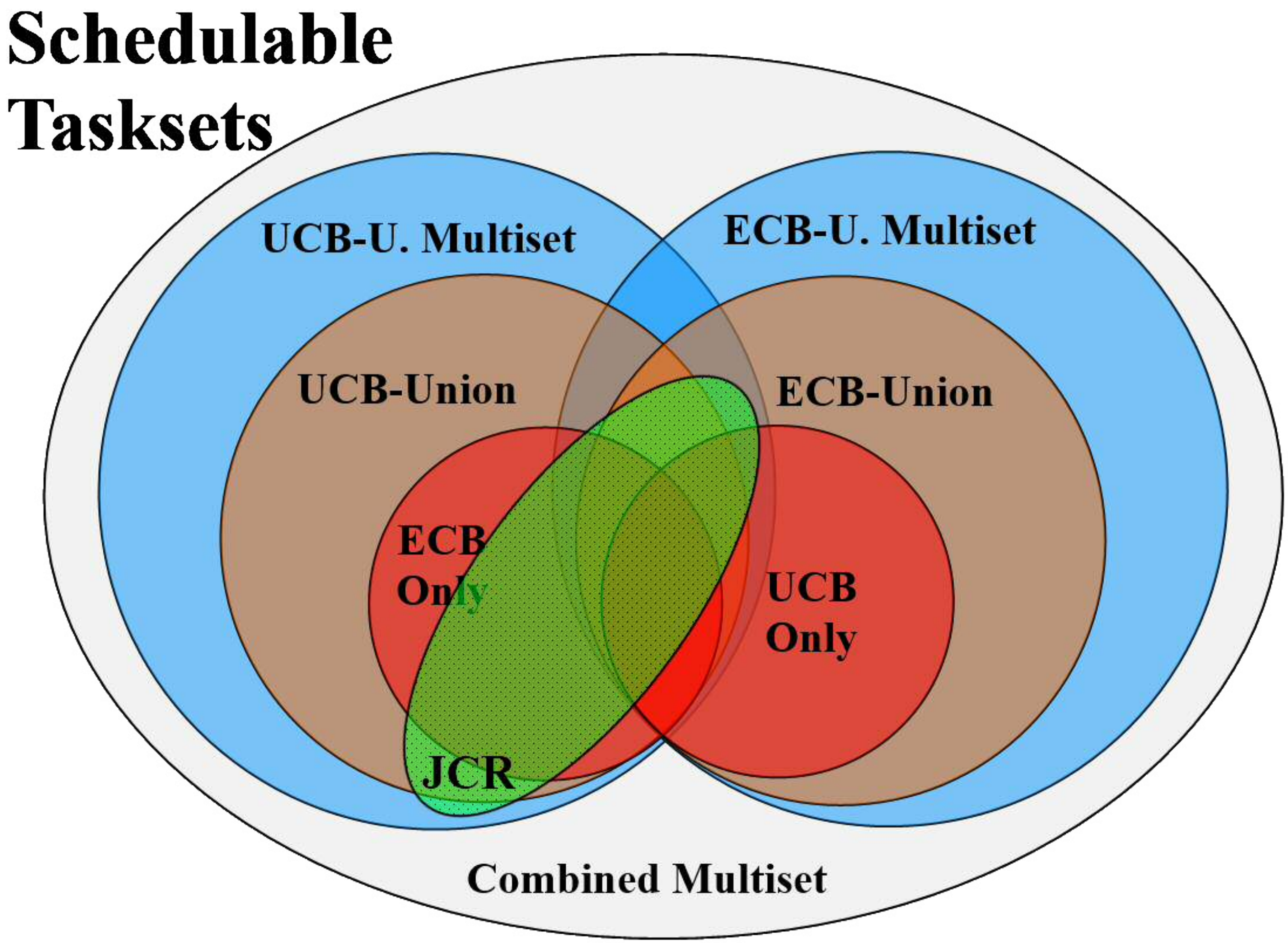
Comparison of approaches



Comparison of approaches



Comparison of approaches



Case study

- Based on a code from the Mälardalen benchmark suite to create a 15 task taskset
- Setup to model an ARM7
  - 10MHz CPU
  - 2KB direct-mapped instruction cache
  - Line size of 8 Bytes, 4 Byte instructions, 256 cache sets
  - Block reload time of 8μs

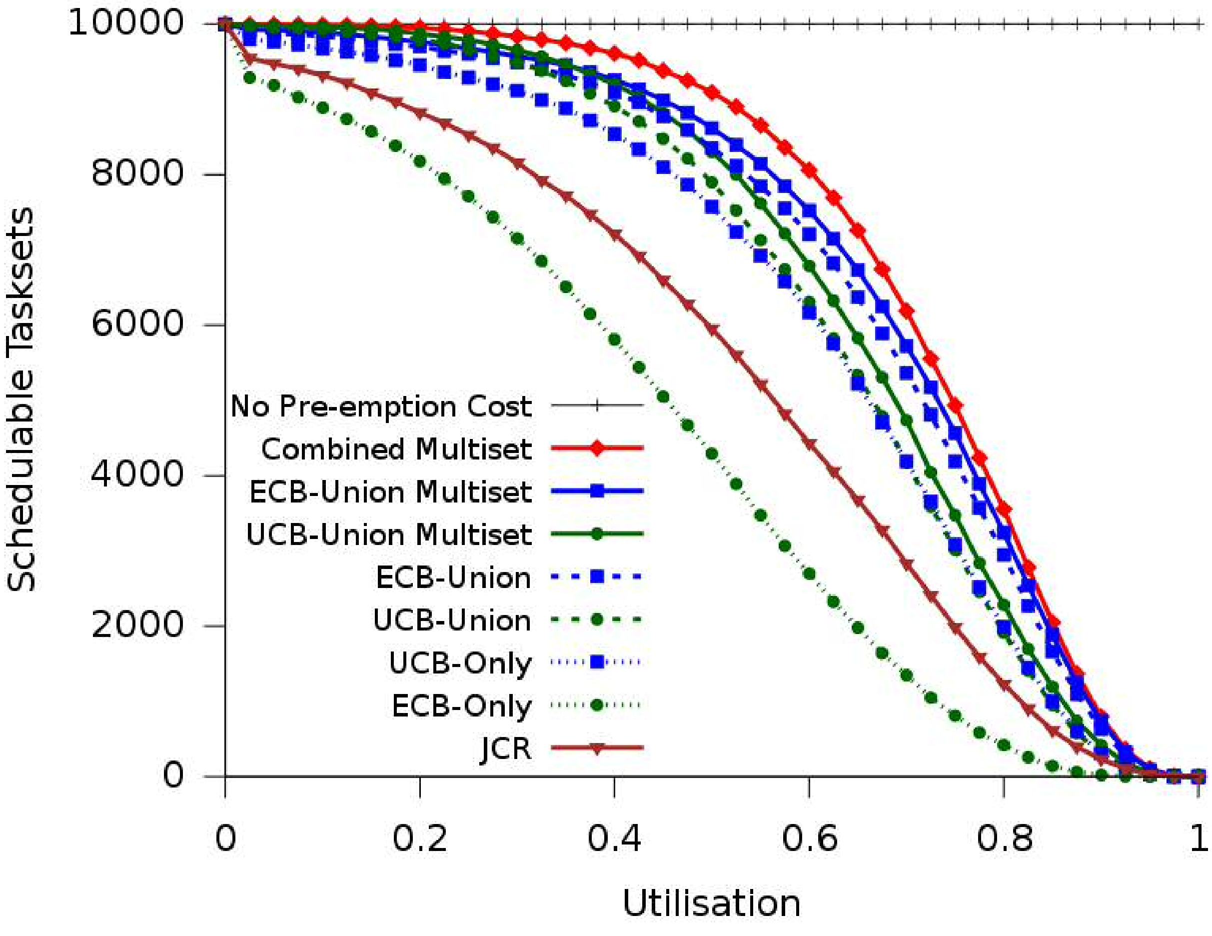
Case study

	Breakdown utilisation
No pre-emption cost	1
Combined Multiset	$\boxed{0.659}$
ECB-Union Multiset	0.659
UCB-Union Multiset	0.594
ECB-Union	0.612
UCB-Union	0.583
UCB-Only	0.462
ECB-Only	0.364
JCR	$\boxed{0.488}$

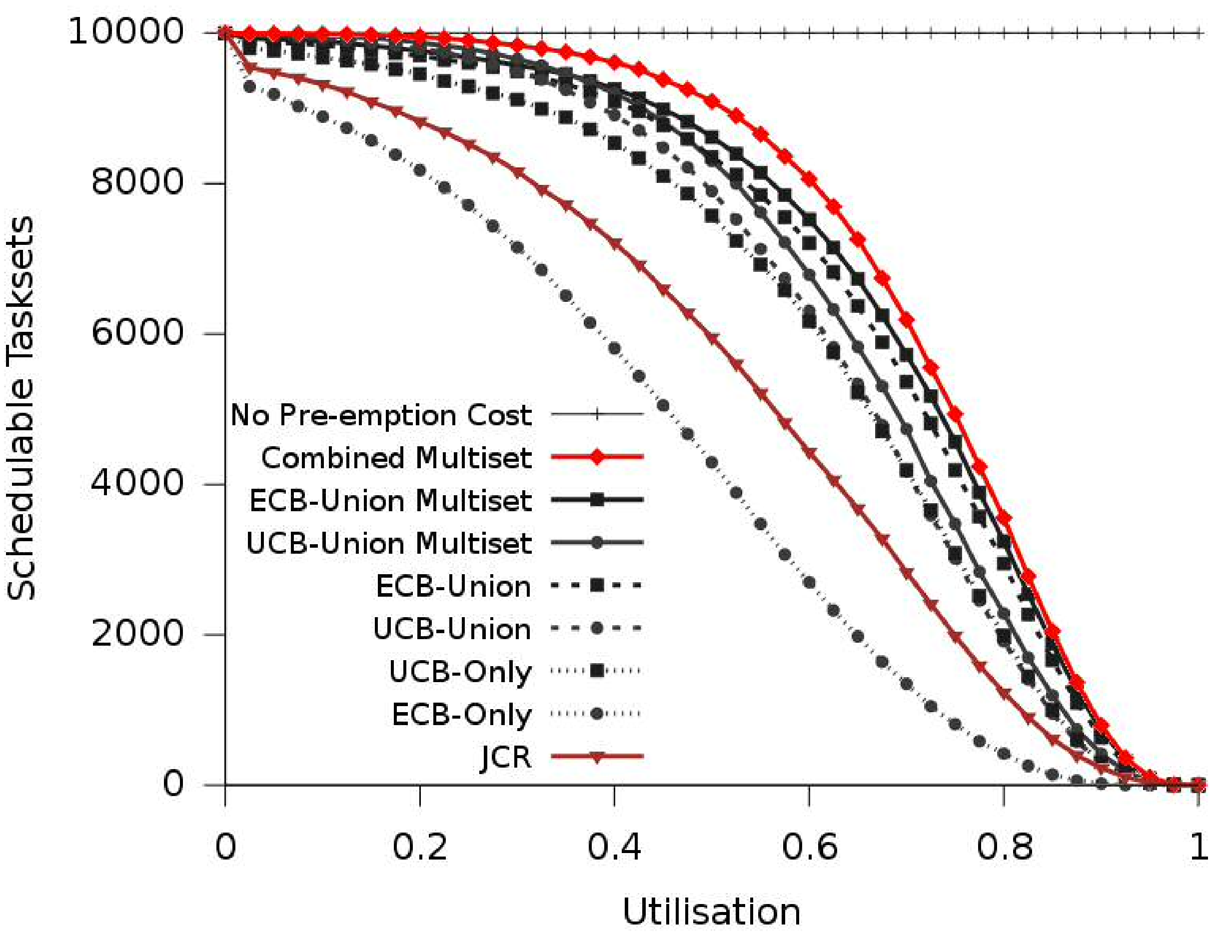
Synthetic tasksets

- 10 tasks per taskset
- 10,000 tasksets for baseline evaluation
- 512 cache sets
- Cache utilisation of 5
- Maximum UCB percentage of 30%

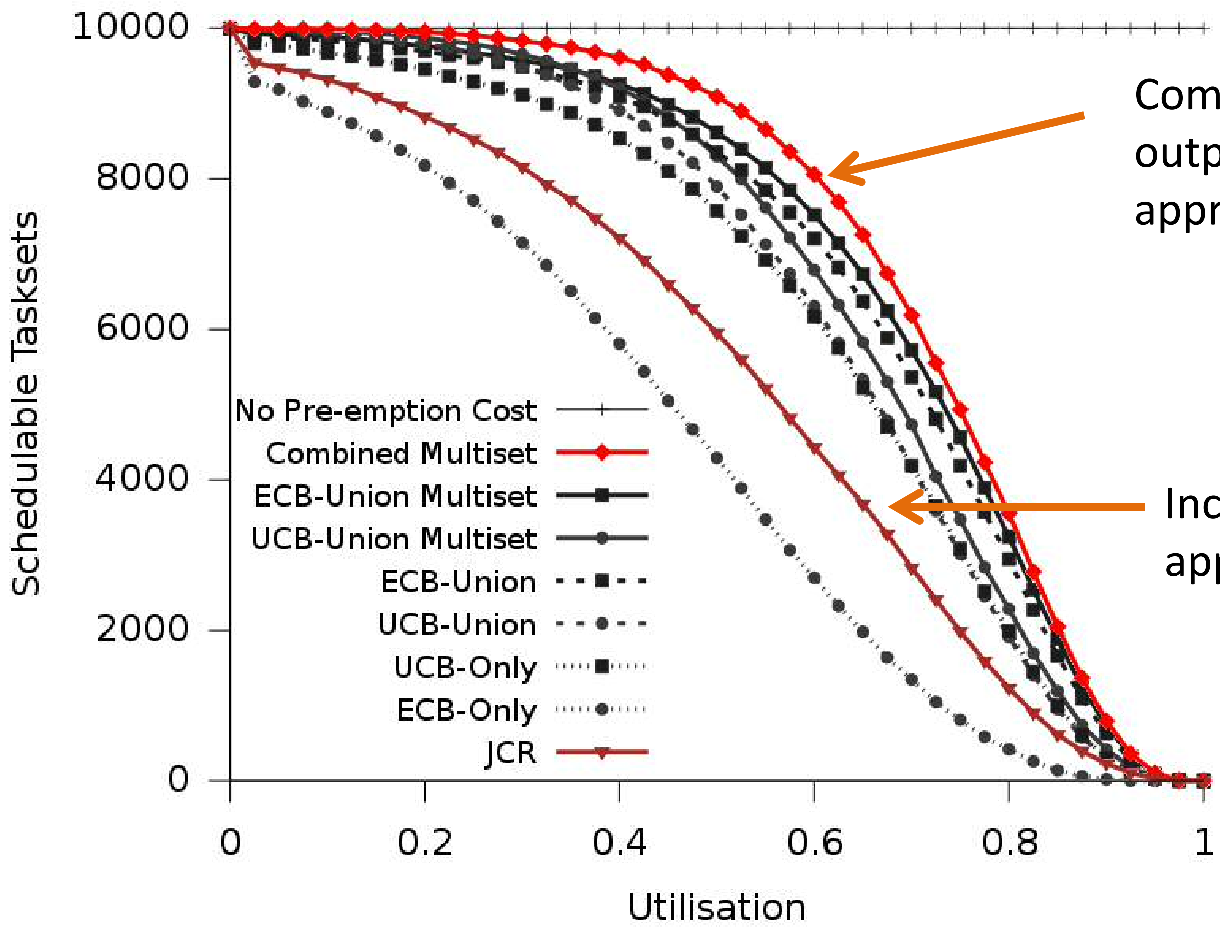
Baseline evaluation



Baseline evaluation



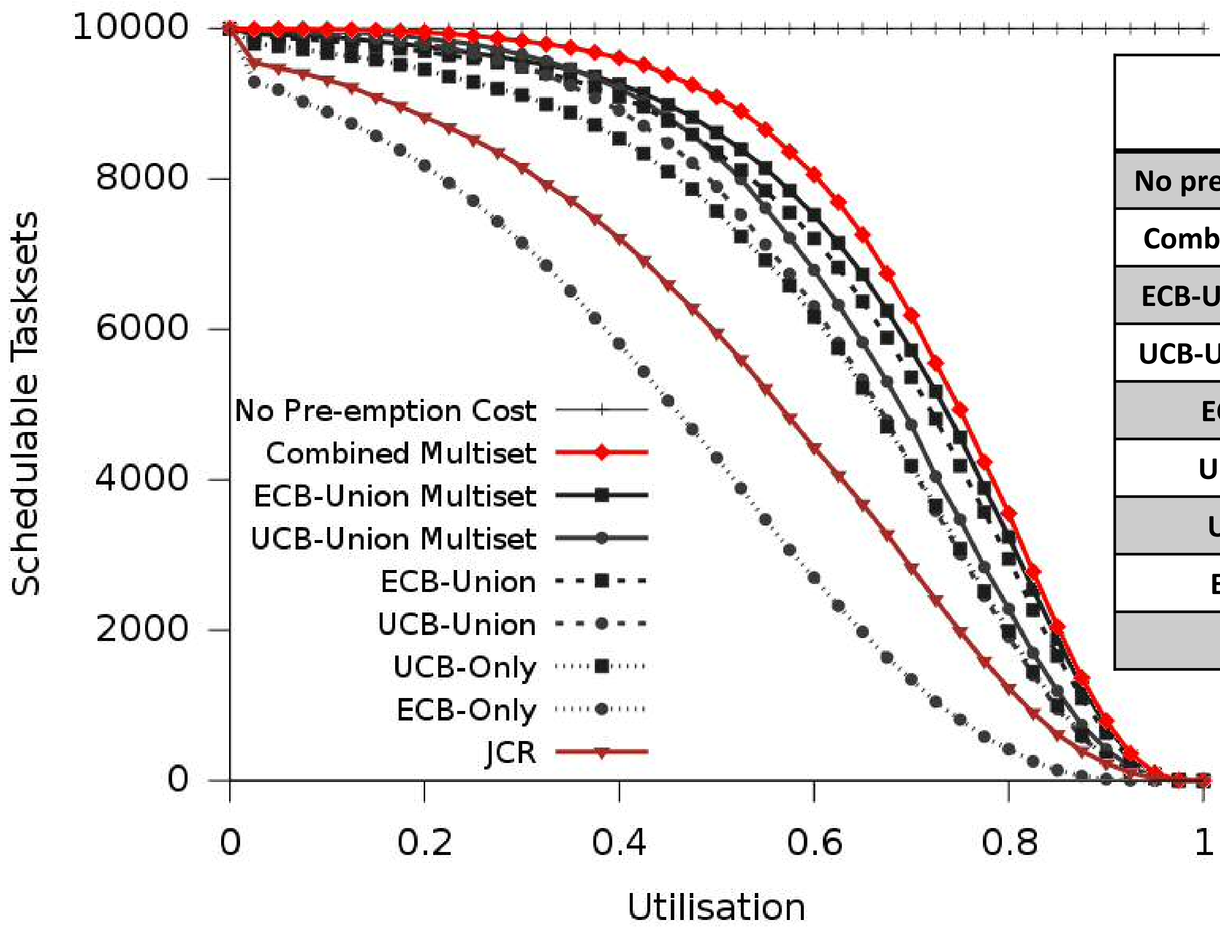
Baseline evaluation



Combined multiset approach outperforms all other approaches

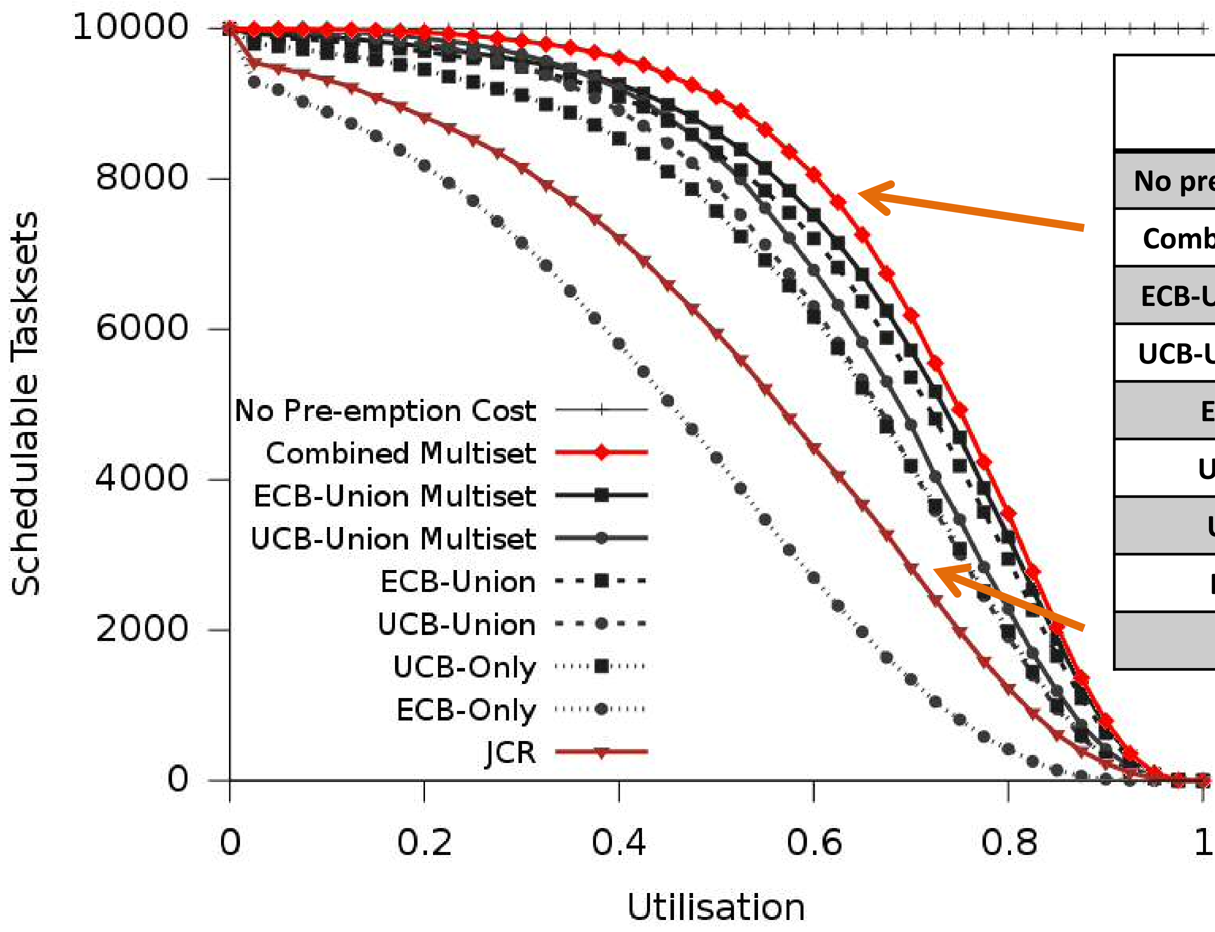
Including the existing approach

Baseline evaluation



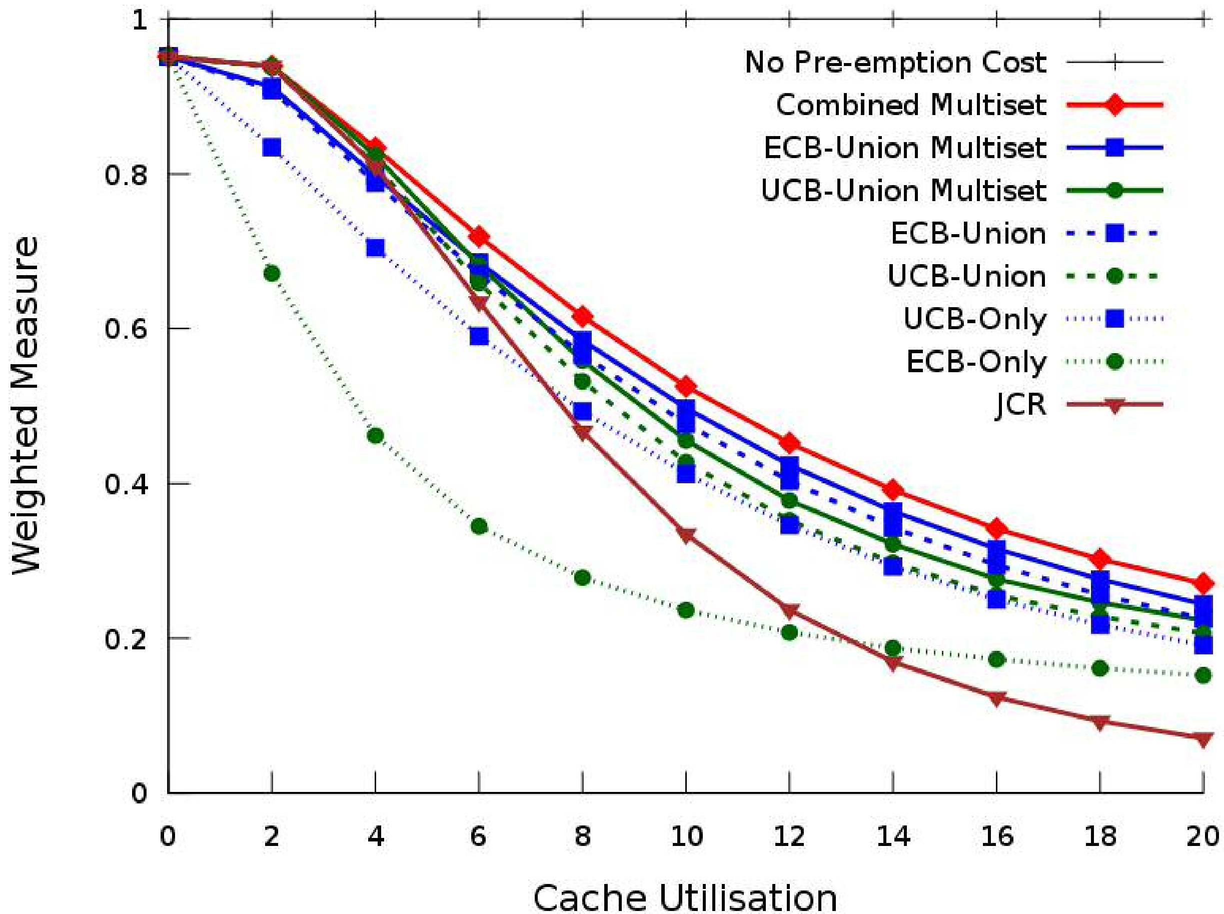
	Weighted Schedulability
No pre-emption cost	1
Combined Multiset	0.528
ECB-Union Multiset	0.501
UCB-Union Multiset	0.455
ECB-Union	0.481
UCB-Union	0.427
UCB-Only	0.416
ECB-Only	0.236
JCR	0.333

Baseline evaluation

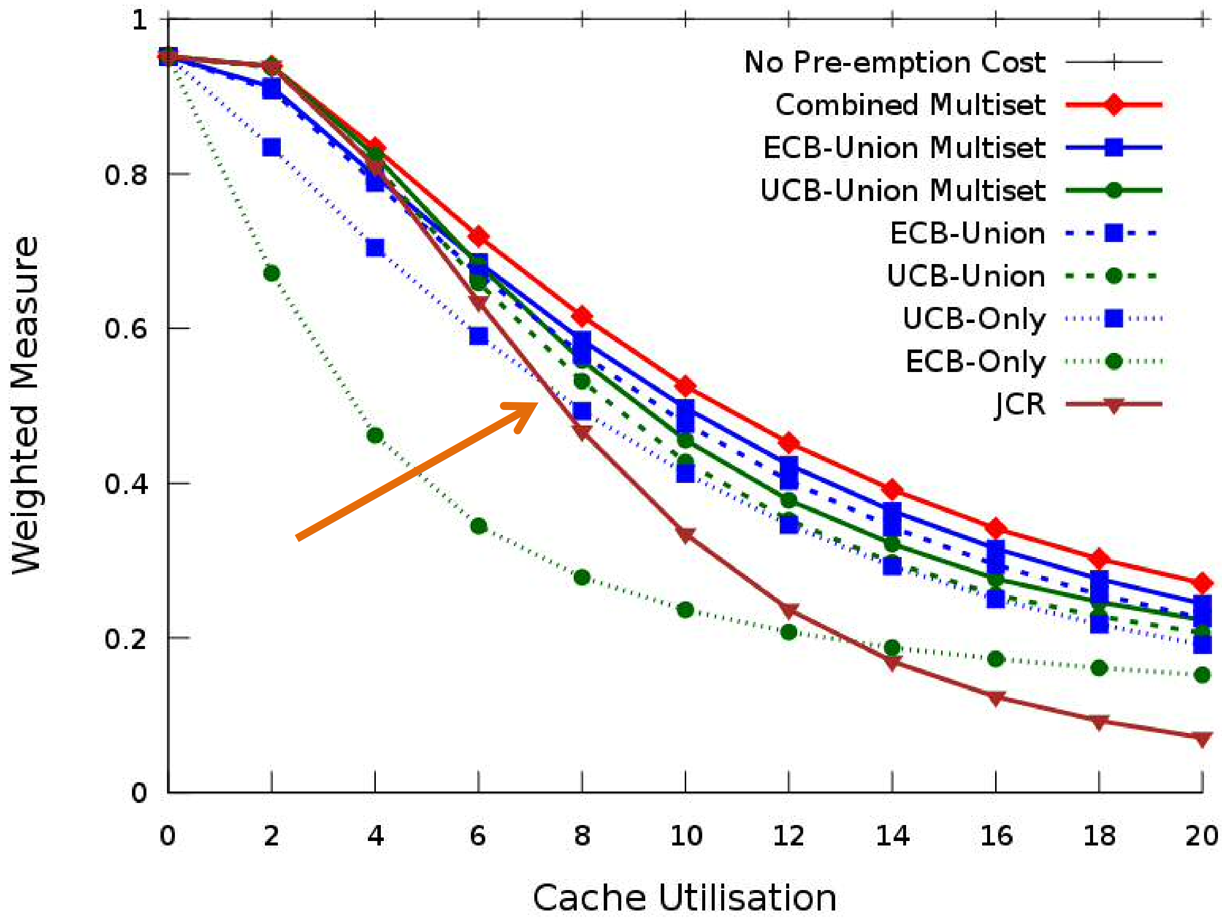


	Weighted Schedulability
No pre-emption cost	1
Combined Multiset	0.528
ECB-Union Multiset	0.501
UCB-Union Multiset	0.455
ECB-Union	0.481
UCB-Union	0.427
UCB-Only	0.416
ECB-Only	0.236
JCR	0.333

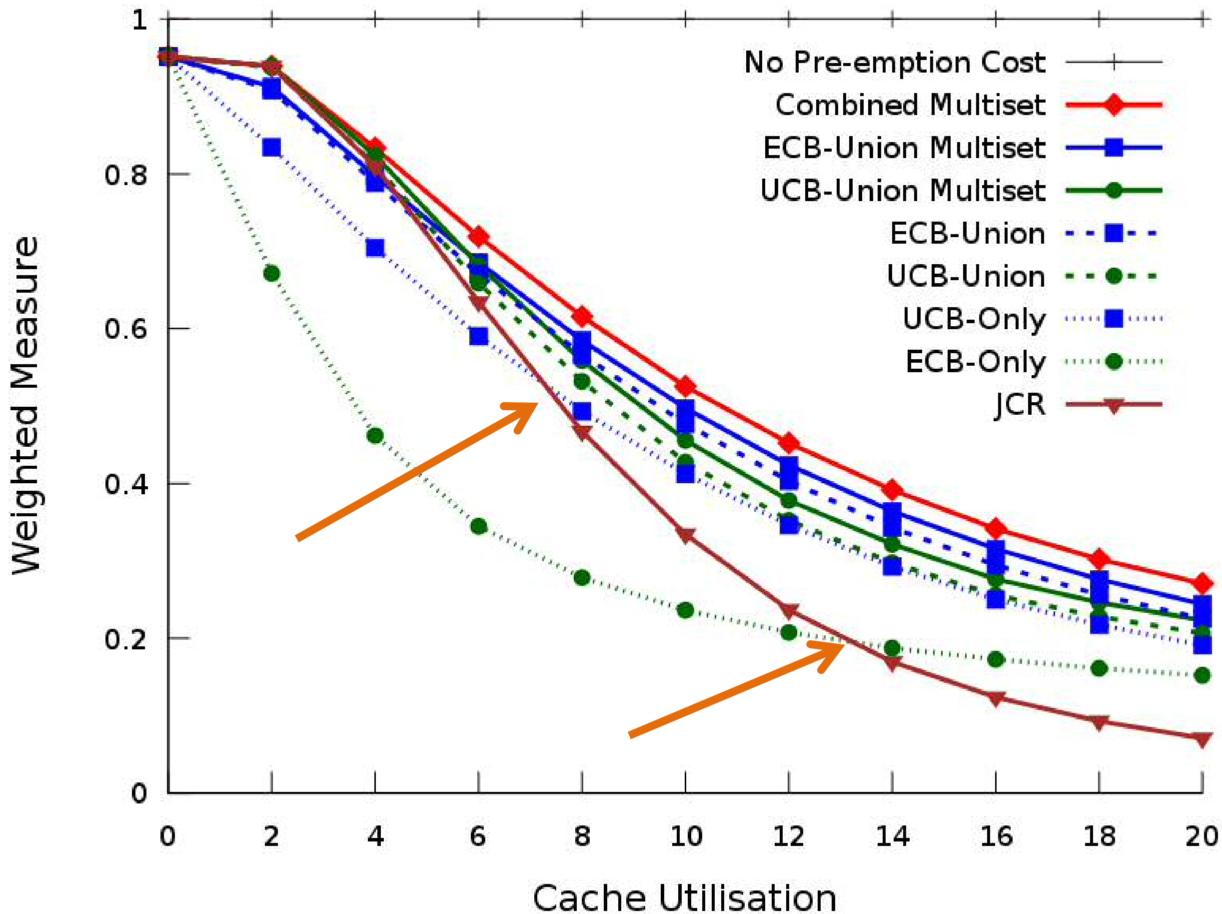
Varying Cache Utilisation



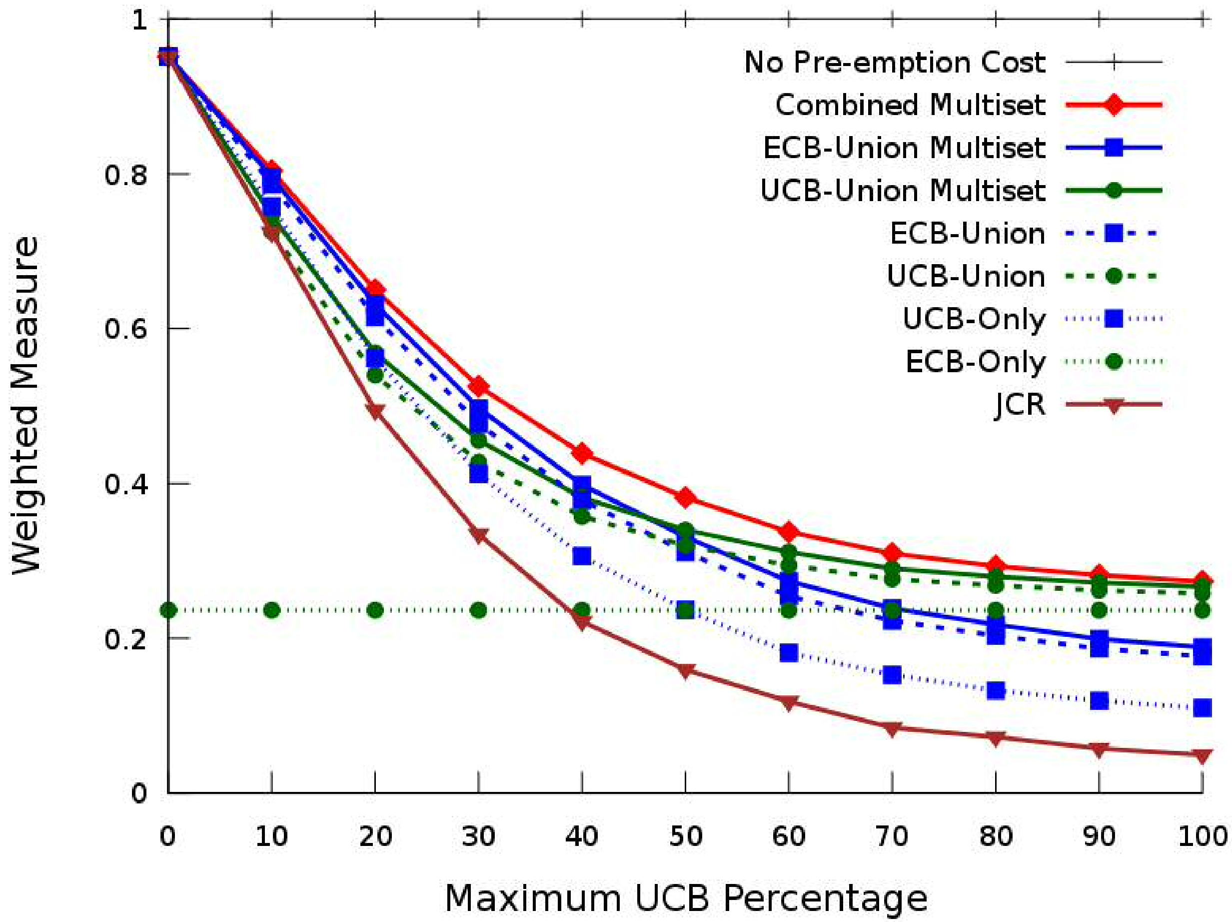
Varying Cache Utilisation



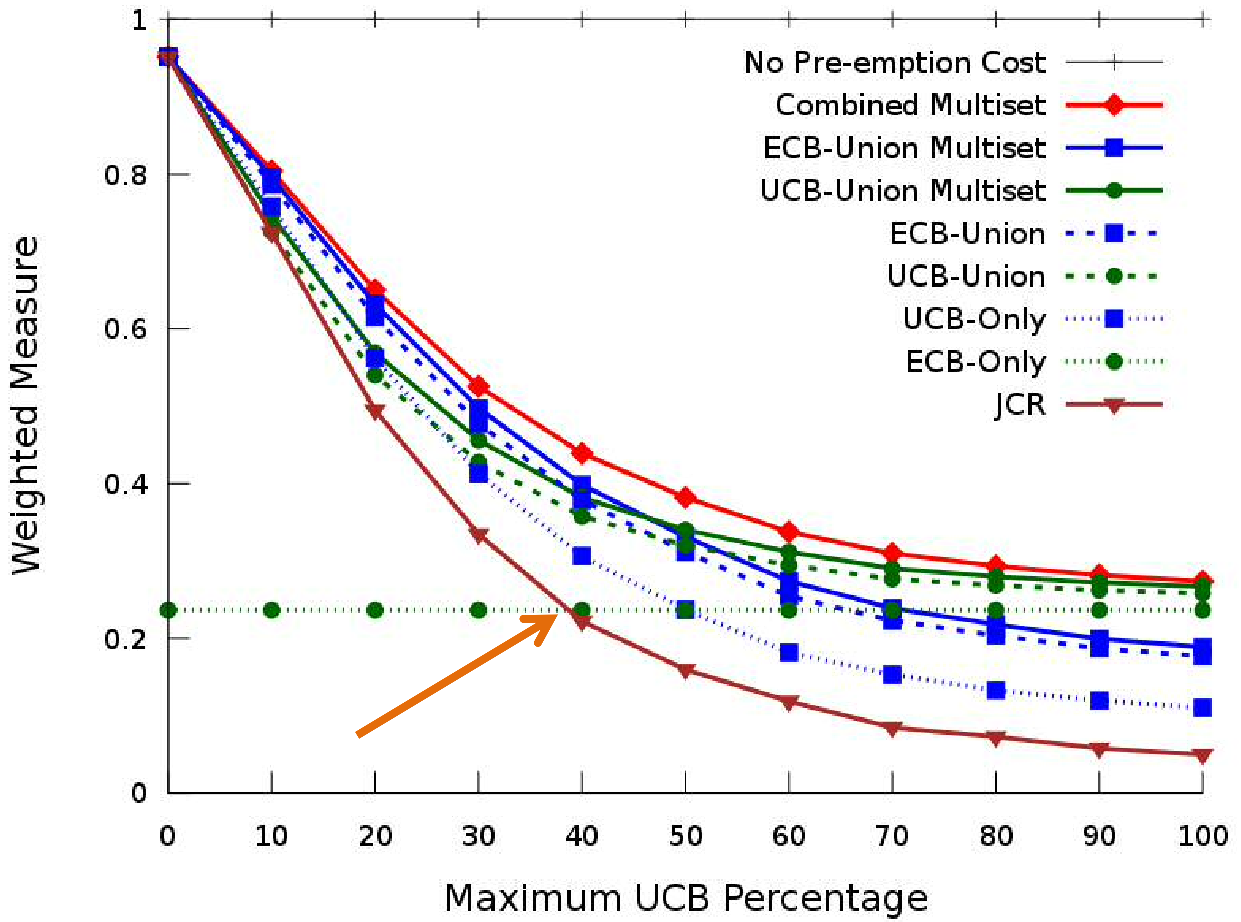
Varying Cache Utilisation



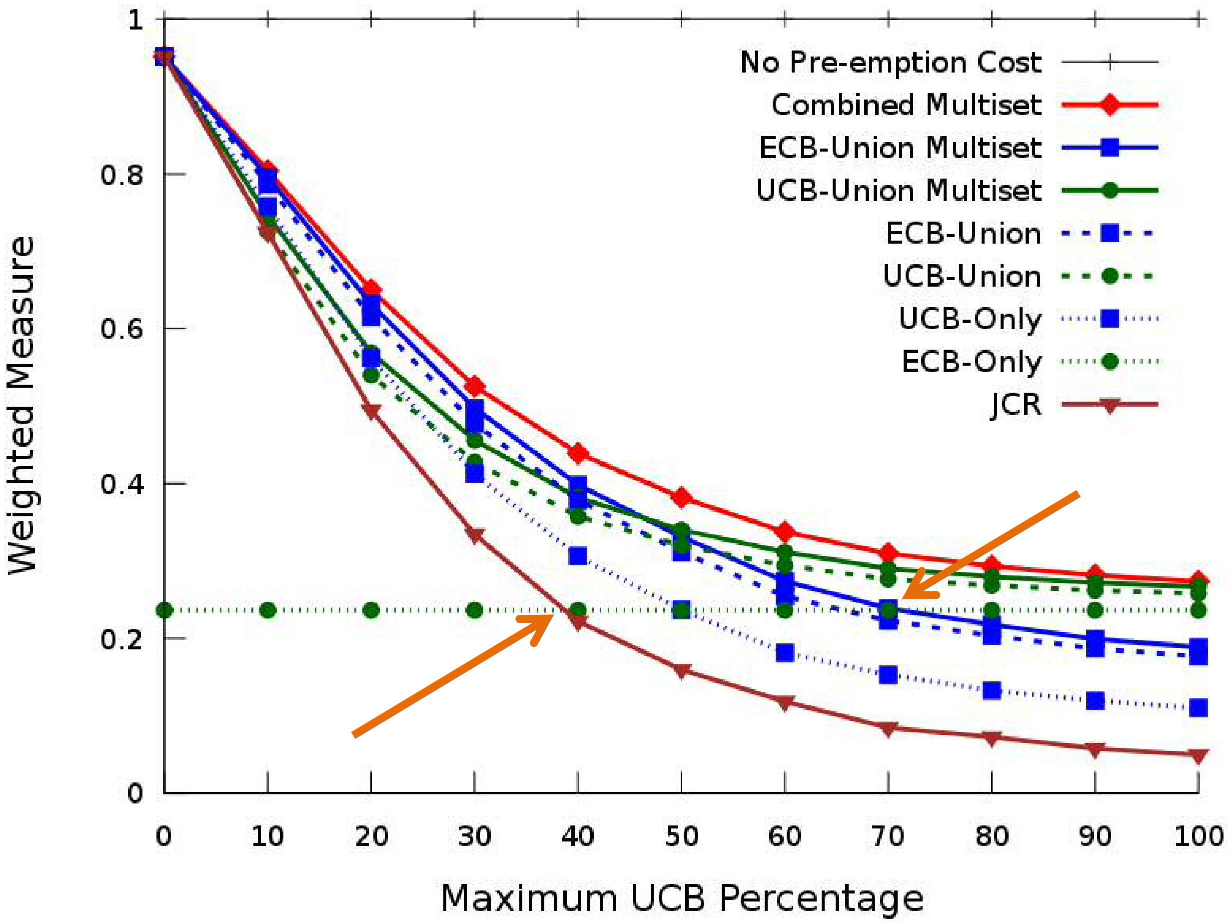
Varying Maximum UCB Percentage



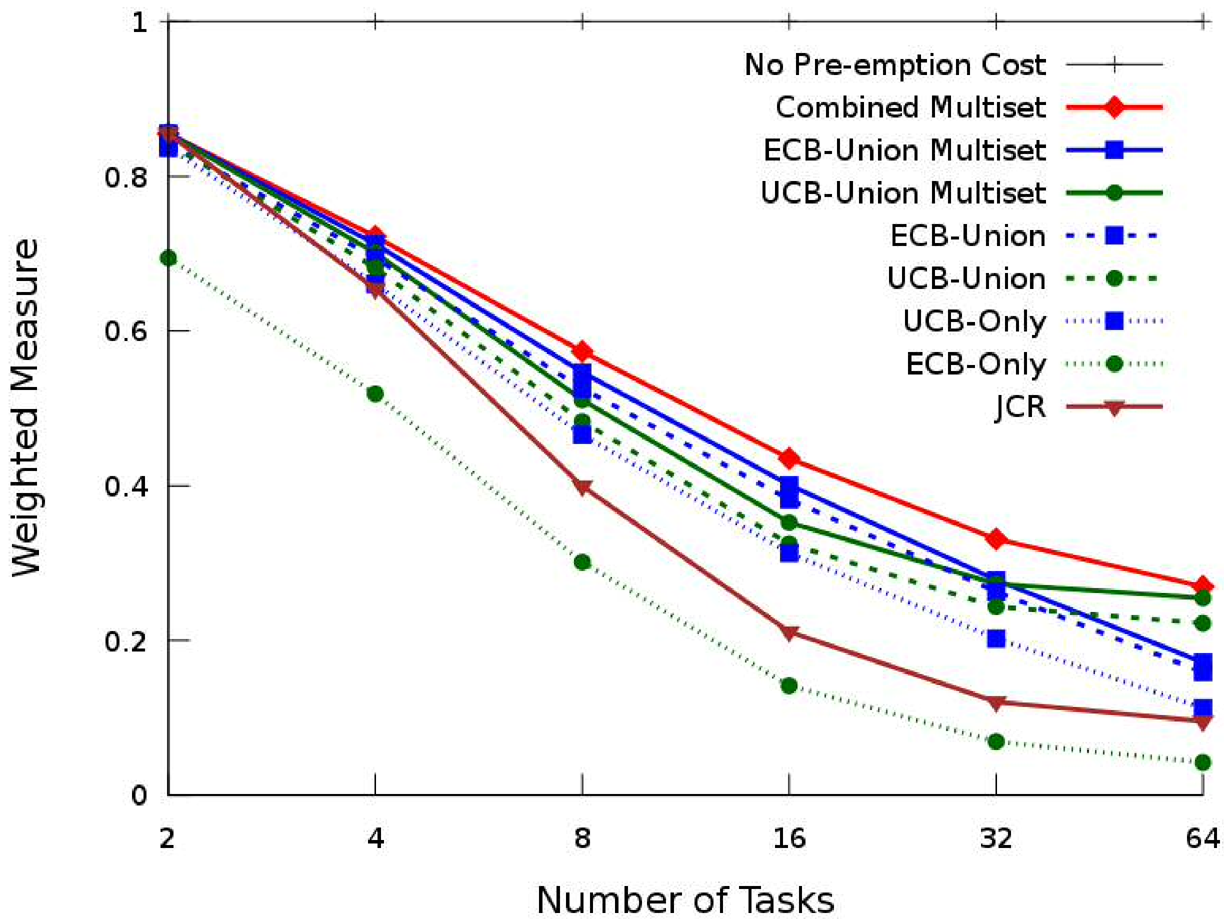
Varying Maximum UCB Percentage



Varying Maximum UCB Percentage



Varying Number of Tasks



Conclusion

- Presented new CRPD aware analysis for EDF
- Combined multiset approach dominates the existing approach by Ju *et al.*
  – Confirmed via evaluation/simulation
- Detailed study shows the strengths and weaknesses of the different approaches
- We plan to investigate which is better, FP or EDF, when taking into account CRPD

Thank you for listening

Any Questions?